(4)

# Implementation and Experimentation with Motion Planning Algorithms

Micha Sharir
Courant Institute of Mathematical Sciences, New York University

Contract Number: N00014-90-J-1284

Final annual Report for the Period 1 Oct 91 – 30 Sep 92
submitted to
The Office of Naval Research

October 1992

1900

Micha Sharir
Courant Institute of Mathematical Sciences, New York University
Phone: (212) 998-3376
E-mail Address: sharir@roband.nyu.edu
Contract Title: Implementation and Experimentation with Motion Planning Algorithms
Contract Number: N00014-90-J-1284
Reporting Period: 1 Oct 91 – 30 Sep 92

## 1 Numerical Productivity Measures

**Refereed papers submitted but not yet published:** 19; 3 others in preparation.

**Refereed papers published:** 3

**Unrefereed reports and articles:** 0

**Books or parts thereof submitted but not yet published:** 1

**Books or parts thereof published:** 0

**Patents filed but not yet granted:** 0

**Patents granted:** 0

**Invited presentations:** 1

**Contributed presentations:** 7

**Honors received (fellowships, technical society appointments, conference committee role, editorship, etc.):** no new honors

**Prizes or awards received (Nobel, Japan, Turing, etc.):** 0

**Promotions obtained:** 0

**Graduate students supported:** 0

**Post-docs supported:** 0

**Minorities supported:** 0

Micha Sharir
Courant Institute of Mathematical Sciences, New York University
Phone: (212) 998-3376
E-mail Address: sharir@roband.nyu.edu
Contract Title: Implementation and Experimentation with Motion Planning Algorithms
Contract Number: N00014-90-J-1284
Reporting Period: 1 Oct 91 – 30 Sep 92

## 2 Detailed Summary of Technical Results

### 2.1 Overview and Background

The main charter of this contract is the implementation and experimentation with motion planning algorithms that emphasize the exact combinatorial and purely geometric approach.

Motion planning is considered to be one of the major research areas in robotics, and is one of the main stages in the design and implementation of autonomous intelligent systems, which is an important long-range goal in robotics research. Motion planning is one of the basic capabilities that such a system must possess. In purely geometric terms, the simplest version of the problem can be stated as follows. The system is given complete information about the geometry of the environment in which it is to operate (and of its own structure), and has to process it so that, when commanded to move from its current position to some target position, it can determine whether it can do so without colliding with any of the obstacles around it, and if so plan (and execute) such a motion.

There are many variants of the problem. A few of those are: motion planning in environments that are only partially known to the system, compliant motion planning that allows contact with obstacles, which might be unavoidable due to measurement errors, optimal motion planning, motion planning with "kino-dynamic" constraints, and motion planning amidst moving obstacles. Still, even the simplest, static, and purely geometric version stated above is far from being simple, and poses serious challenges in the design of efficient and robust algorithms.

Theoretical studies of motion planning have been abundant in the past decade, and the Principal Investigator has been involved with many of them. It was shown that the main parameter that controls the computational complexity of the problem is the number $k$ of degrees of freedom of the system. When $k$ is arbitrarily large (e.g. in coordinated motion planning of many independent bodies), the problem usually becomes computationally intractable [9,10]. There are several general techniques (one by Schwartz and Sharir [17] and a more recent and more efficient one by Canny [5]) that have been derived for solving the problem for arbitrary systems, but their worst case running time is exponential in $k$, and even for available commercial systems with $k = 6$ degrees of freedom, these algorithms are very complex and unacceptably inefficient for practical use.

This situation has caused subsequent research to proceed in two divergent directions. One was to abandon the exact algorithmic approach and design heuristic and approximating techniques in which the geometry of the space of available placements of the system is not computed accurately but only coarsely approximated, or is "bypassed" by other heuristic techniques. The resulting systems are generally not robust — they might miss free motions and declare incorrectly the desired destination as unreachable. Moreover, even with the heuristic shortcuts, these systems are still inefficient, and most of them perform well (within the above mentioned limitations) only on 'toy' examples consisting of only a few obstacles.

The other approach was to continue to cling to the exact combinatorial algorithmic paradigm, but begin by attacking problems with a small number of degrees of freedom, analyze them thoroughly, and develop efficient algorithms whose worst-case running time is even better than that of the general technique of Canny. This approach, which is the one followed in our research, is a 'bottom-up' approach, that aims to solve simpler systems first, in the hope that these solutions will be usable as routines in the solutions of more general problems. Moreover, this approach leads to better understanding of the combinatorial structure of the space of free system placements, and can therefore result in solutions that are faster than those yielded by heuristic techniques.

Although many motion planning problems with very few degrees of freedom are not very realistic, some of them do correspond to problems that can arise in practice. For example, the problem involving a rigid polygonal object moving in the plane amidst a collection of polygonal obstacles is actually the problem of navigating a robot vehicle, and has only three degrees of freedom. Navigating a circular robot has only two degrees of freedom. These problems have been successfully attacked by the exact algorithmic technique, and a battery of efficient techniques for their solutions has been developed (see [16], [11], [14], [12], [8]). Some of these solutions have in fact been implemented and tested (see e.g. [13], and also [4]), although no real production system has resulted from these experiments, as far as we know.

In the present research we have chosen another class of problems involving three degrees of freedom and have the potential of being applicable in real-life problems. This class involves a rigid object flying through 3-dimensional space, by translation only, amidst a collection of polyhedral obstacles (which are static, and whose geometry is known to the system, as in our basic assumptions made above). In full generality, the flying motion of a rigid object in 3-space involves 6 degrees of freedom (with rotation) and is too complex, in the present state of the field, for exact and practical algorithmic solution. The case of allowing only translations can still be used in practice in several ways: (i) If the size of the moving object is much smaller than the sizes of the obstacles, we can approximate the object by a point, which has only the three degrees of freedom of translation. (ii) If the moving object has a generally rounded shape, we can approximate it by a moving ball, again with only three degrees of freedom. (iii) We can use the solution for translational motion planning to obtain an approximate solution of the general problem, by discretizing over the range of available orientations, solve the purely translational problem for each orientation, and look for purely rotational passages between adjacent orientations; this technique has been recently proposed for planar motions in [1], and it seems applicable to 3-dimensional problems as well.

This problem has already been discussed in a pioneering paper on algorithmic motion planning [15] 11 years ago. However, no analysis, nor even any consideration of algorithmic efficiency, has been provided there. Recently the problem has been studied and analyzed in several papers. The case of a moving point has been studied in [6]. It was shown there that if the polyhedral obstacles consist of $n$ faces and $r$ convex edges (that is reflex edges from the point of view of free space), then the free space can be decomposed into $O(n + r^2)$ tetrahedra, in time $O(nr + r^2 \log n)$. Having this decomposition available in the form of a 'connectivity graph', whose vertices are these tetrahedra and whose edges connect pairs of adjacent tetrahedra, facilitates a reduction of the motion planning problem to a simple (and discrete) path searching problem in that graph. The solution given in [6] is slightly complicated and requires the use of a few sophisticated algorithmic techniques. A generalization of the problem to the case of an arbitrary translating polyhedral object has been studied in [2], which showed that the complexity of a single connected component of the free configuration space is at most $O(n^{7/3})$, which is a significant improvement over the naive (and worst-case tight) $O(n^3)$ bound on the complexity of the entire configuration space. A *major theoretical breakthrough* of our research in the past year is an improvement of this bound to $O(n^2 \log n)$ [3]. Note that a single component of the free configuration space, namely the one that contains the starting position of the robot, is all we need, because all placements reachable from this starting position must necessarily lie in that component. The paper [2] also presents a randomized algorithm to compute a single component in time that is close to $O(n^{7/3})$. By plugging our improved complexity bound into this algorithm, one can show that its running time drops down to close to $O(n^2)$, which constitutes a significant and near-optimal result, since in the worst case the complexity of a single cell can indeed be quadratic.

## 2.2 System Description

The implementation of our 3-d motion planning system has been carried out by a full-time programmer (Ms. Estarose Wolfson) at the Robotics Lab of the Courant Institute at New York University. Currently, the implementation of the simple case of a moving point has been completed, and extensive testing of it has been conducted. In addition, we have developed a graphics output package that displays varioue aspects of the system output on a Silicon Graphics IRIS workstation. Using this package we have produced a demo videotape illustrating the performance of the system, and the figures shown below display some snapshots of this videotape.

A detailed description of our system was given in last year's report, and we repeat it here in somewhat condensed form, which also includes the illustration of the performance of the algorithm by several output figures. A full report on our system is given in the technical report [18] that we are currently preparing.

A major principle in the system design was to implement a system whose worst case running time matches the best available theoretical solutions (in our case, that of [6]), but to trade sophisticated algorithmic techniques by simpler methods whenever possible (without hurting the overall asymptotic running time). For example, consider the spatial *point location* problem, which arises a lot in our implementation. A simple version of the problem asks to determine, for an arbitrary query point, the obstacle face it 'sees' directly above it. There are several recent efficient techniques for solving this problem, but they are very cumbersome and practically inefficient to implement. In our system we used a simpler solution that proceeds by traversing faces of the obstacles in a certain order until the one lying directly above the point is found. This method is very simple to implement, and its total cost turns out in our case to be within the allowed theoretical bound. This policy has been followed in all other steps of our algorithm.

Here is a brief sketch of the structure of our system (see also last year's report).

**OBJECTIVES and TERMINOLOGY:** Given any two points in 3-space and a set of polyhedral obstacles having a total of $N$ faces, we wish to determine whether there is a path between these points (avoiding intersections with the obstacles) and if so find one such path. This is the motion planning problem of moving a point through a three dimensional space consisting of non-overlapping obstacles. To do this, the complementary space of the obstacles (with respect to some large imaginary enclosing box), called the *free space*, is decomposed into convex units (cells), which form the nodes of a *connectivity graph* whose edges connect pairs of adjacent cells. These cells have walls consisting of $z$-vertical planar faces and top and bottom 'covers' each consisting of facets from a unique obstacle. Thus these basic cell units and the connectivity among them will allow us to travel through free space to reach our destination, provided it lies in the same connected component of free space as our initial position (which is the same as belonging to the same connected component of the connectivity graph).

The general technique, as developed in [6], [2], and others, is to construct a *vertical cell decomposition* of the free space. Such a decomposition is obtained by erecting vertical walls up and down from each *reflex* obstacle edge (i.e. an edge whose dihedral angle within the free space is greater than 180 degrees). These walls are extended until they hit other obstacle faces (or, failing this, to infinity). Collectively, they partition free space into convex subcells of the form discussed above, and their adjacency through the vertical walls gives us the desired connectivity graph.

We have modified this method so that walls are erected only from *full reflex* edges, which are edges $e$ with the property that the vertical plane passing through $e$ is such that the obstacle containing $e$ lies (locally) only on one side of the plane. This coarser decomposition yields cells that are only "$z$-convex", meaning that any $z$-vertical line intersects such a cell in a connected segment. It is still relatively easy to navigate through such a cell, and in practice the saving in this coarser scheme is expected to be significant. We denote by $r$ the total number of reflex edges and by $R$ the number of full reflex and *inverse reflex* edges (defined in analogy to reflex edges except that the free space lies locally on one side of the vertical plane through the edge).

A key concept in our method is that of obstacle *silhouettes*, which are loci of points on the obstacle boundaries where the $z$-vertical cross section of the obstacle has a discontinuity. Such a silhouette consists of a connected closed cycle of full reflex obstacle edges, inverse reflex edges, or a combination of such edges.

The silhouettes contain most of the information necessary to achieve our coarse cell decomposition, and the total size of all silhouettes is only proportional to $R$ and not to $N$, again implying significant savings in practice (and theory).

In addition, we use the notion of *half reflex* edges, which are all the remaining reflex edges, whose two adjacent faces lie on opposite sides of the vertical plane passing through the edge. Thus, if we were to erect vertical walls from such an edge (which we do not), the wall would extend only upwards or only downwards into free space. Half reflex edges are used in the later stages of the program to plan passages through the resulting cells.

The input to the system consists of the obstacles. These are arbitrarily complex 3-d polyhedra, that may have holes, tunnels, handles, etc. We assume that they are given by their boundary representation, where each face is already triangulated, and comes with its outward-directed normal vector.

## METHOD and PROGRAM:

For lack of space, we only give a very brief outline of the system.

(1) We calculate the obstacle silhouettes by a breadth first search on the vertices and edges of each obstacle, and connect them locally into appropriate lists.

(2) We next find the "critical points" of the silhouette interactions by performing a planar sweep along the $x$ direction on the $xy$-projections of the silhouettes. The critical points are the $x$ minimum and maximum points of the branches of the silhouettes, the midpoints where tunnel holes change from being inside to outside the obstacle (inverse to reflex edges of silhouette) and vice-versa, and the intersection points of two projected silhouettes whose obstacles are adjacent in the $z$-direction of 3-space. The running time of this stage is $O(N + (R + S)\log X)$, where $X < R$ is the number of chains and $S < R^2$ is the number of intersections between them.

(3) For each cell silhouette we complete the construction of the $z$-vertical walls erected from the silhouette edges. For this we need to find their top and bottom intersections with the obstacles by 'tumbling' along the path of the chain of edges of the silhouette from one critical point to another, knowing that between any two critical points the $z$ neighbor above (and below) the edge will remain on the same obstacle patch;

(4) We are now in a position to actually construct our cells and the connectivity between them. We split each chain of reflex edges at its critical points, and then recombine the resulting chain fragments (and the vertical walls attached to them) to form the contours of new coherent cells. The recombination is done locally around each critical point, by attaching chain fragments and their walls to adjacent fragments-and-walls meeting them at this point, and by determining locally the geometry of the resulting incident cells.

(5) The cells just produced are "$z$-convex" — any vertical line intersects such a cell in a connected interval, but their $xy$-projections can still have an arbitrary polygonal shape. The next step decomposes our cells further into "more convex" subcells, each being $z$-convex and having a convex $xy$ projection. This is achieved by an appropriate planar sweep through the $xy$ projection of each cell, and can be done in total time $O(NR)$.

(6) Next we find certain actual paths through the cells. These paths connect some center point within each cell to entry / exit points on the vertical walls separating the cell from adjacent ones. To do this, we pass a vertical plane through the center point $p$ and some entry/exit point $q$, and trace the intersections of this plane with the top and bottom covers simultaneously, using our tumbling method. Our strategy is to remain always at mid-height between the current top and bottom faces. We thus obtain a polygonal path whose $xy$-projection is a striaght segment. We collect all these paths and store them in a data file, to be used by the final motion planning phase. With some care, the cost of this step can also be made $O(NR)$.

(7) **The Motion planning phase:** Finally, given a source point $p$ and a target point $q$, we want to determine whether there exists a free path between $p$ and $q$, and, if so, produce such a path. For each of the points $p$, $q$, we find the cell containing the point or indicate that the point is not in free space (in which case no motion planning has to be done). For points in free space, let $c_1$, $c_2$ denote the cells containing $p$ and $q$ respectively. We also find the path from $p$ to the center of $c_1$ and from the center of $c_2$ to $q$. We next test if these two cells are in the same connected component of our connectivity graph. If this is the case, we find a path in the graph connecting $c_1$ and $c_2$ by a simple breadth first search. We then construct the actual path from $p$ to $q$ by concatenating the subpaths from $p$ to the center

of $c_1$, from the center of each cell to an exit point on the vertical wall separating it from the next cell, from that point to the center of the next cell, and finally from the center of $c_2$ to $q$. The output of this phase is simply a sequence of points, given by their coordinates, so that between any two consecutive points the path proceeds along a straight segment. The running time of this step, and the size of the output path, are both $O(N + R^2)$.

(8) **Graphical output:** We have implemented a graphical output package that reads the files produced by our system and transforms them into several video output sequences, displayed on a Silicon Graphics IRIS workstation at our Lab. The output shows the given polyhedral scene, where the obstacles are shown both in solid form and in triangulated wire-frame form, and where the initial and final placements of the moving robot are highlighted. The output collision-free path planned by the system is shown, and then the viewer is "taken on a ride on the robot" as it moves along the path. During this ride, the view of the current cell is constantly displayed, where the vertical boundaries of the cell are shown as cage-like bars; each time we cross through such a boundary, the former cell vanishes and is replaced by the new cell we are entering. The figures given below show snapshots of the video produced by this package. This video output has been used to create a demo videotape of our system, which is enclosed with this report.

## 2.3 Supplemental Theoretical Research

Besides work on the system proper, we have also continued to work on related problems in motion planning and in computational geometry. Some parts of this work are closely relevant to the research project, while other parts cover more basic problems in computational geometry. The main result related to the research project is the improved bound, already mentioned above, on the complexity of a single cell in an arrangement of triangles in 3-space, which in turn leads to an efficient algorithm for motion planning of the type we study in the project. Among our other results that are more relevant to robotics, we mention: improved bounds and efficient algorithms for certain motion planning problems with three degrees of freedom, analysis of the complexity of the union of polyhedra in space, upper envelopes of Voronoi surfaces and their applications in pattern recognition, optimal placement problems of polygons in a polygonal environment, computing a single face in an arrangement of line segments, and some extensions of this algorithm, a note on an earlier motion planning algorithm, and miscellaneous results in computational geometry, including efficient techniques for ray and circle shooting in polygonal regions, improved techniques for output-sensitive hidden surface removal, geometric location and other optimization problems, a new randomized algorithm for linear programming and its analysis, and applications of a new space partitioning technique. A bibliography of publications that acknowledge support by the grant (in which these works appear) is given in Section 3 below.

## References

[1] H. Alt, R. Fleischer, M. Kaufmann, K. Mehlhorn, S. Näher, S. Schirra and C. Uhrig, Approximate motion planning and the complexity of the boundary of the union of simple geometric figures, *Proc. 6th ACM Aymp. on Computational Geometry,* 1990, pp. 281–289.

[2] B. Aronov and M. Sharir, Triangles in space, or building (and analyzing) castles in the air, *Combinatorica* 10 (2) (1990), 137–173.

[3] B. Aronov and M. Sharir, Castles in the air revisited, *Proc. 8th ACM Symp. on Computational Geometry,* 1992, 146–156.

[4] F. Avnaim, J.D. Boissonnat and B. Faberjon, A practical exact motion planning algorithm for polygonal objects amidst polygonal obstacles, *Proc. IEEE Symp. on Robotics and Automation,* 1988, pp. 1656–1661.

[5] J. Canny, *The Complexity of Robot Motion Planning,* Ph.D. Dissertation, M.I.T., 1987.

[6] B. Chazelle and L. Palios, Triangulating non-convex polyhedra, *Discrete Comput. Geom.* 5 (1990), 505–526.

[7] H. Edelsbrunner, L. Guibas and J. Stolfi, Optimal point location in a monotone subdivision, *SIAM J. Computing* 15 (1986), 317–340.

[8] L. Guibas, M. Sharir and S. Sifrony, On the general motion planning problem with two degrees of freedom, *Discrete Comput. Geom.* 4 (1989), 491–521.

[9] J.E. Hopcroft, D.A. Joseph and S.H. Whitesides, Movement problems for 2-dimensional linkages, *SIAM J. Computing* 13 (1984), 610–629.

[10] J. Hopcroft, J.T. Schwartz and M. Sharir, On the complexity of motion planning for multiple independent objects: PSPACE-hardness of the 'warehouseman's problem', *Int. J. Robotics Research* 3 (4) (1984), 76–88.

[11] K. Kedem, R. Livne, J. Pach and M. Sharir, On the union of Jordan regions and collision-free translational motion amidst polygonal obstacles, *Discrete Comput. Geom.* 1 (1986), 59–71.

[12] K. Kedem and M. Sharir, An efficient motion planning algorithm for a convex rigid polygonal object in 2-dimensional polygonal space, *Discrete Comput. Geom.* 5 (1990), 43–75.

[13] K. Kedem and M. Sharir, Implementation and experimentation with an efficient motion planning algorithm for a convex polygon in 2-d polygonal space, *Proc. 4th ACM Aymp. on Computational Geometry*, 1988, 329–340.

[14] D. Leven and M. Sharir, Planning a purely translational motion for a convex object in two-dimensional space using generalized Voronoi diagrams, *Discrete Comput. Geom.* 2 (1987), 9–31.

[15] T. Lozano-Perez and M. Wesley, An algorithm for planning collision-free paths among polyhedral obstacles, *Comm. ACM* 22 (1979), 560–570.

[16] C. Ó'Dúnlaing and C.K. Yap, A 'retraction' method for planning the motion of a disc, *J. of Algorithms* 6 (1985), 104–111.

[17] J.T. Schwartz and M. Sharir, On the piano movers problem: II. General techniques for computing topological properties of real algebraic manifolds, *Adv. Appl. Math.* 4 (1983), 298–351.

[18] E. Wolfson and M. Sharir, Design and implementation of a motion planning system in 3-space, in preparation.

Micha Sharir
Courant Institute of Mathematical Sciences, New York University
Phone: (212) 998-3376
E-mail Address: sharir@robard.nyu.edu
Contract Title: Implementation and Experimentation with Motion Planning Algorithms
Contract Number: N00014-90-J-1284
Reporting Period: 1 Oct 91 – 30 Sep 92

## 3 Lists of Publications, Presentations, Reports, and Honors

### 3.1 Publications

**Includes papers listed in last year's report whose status has changed**

1. D. Halperin and M. Sharir, Improved combinatorial bounds and efficient techniques for certain motion planning problems with three degrees of freedom, *Computational Geometry, Theory and Appls* 1 (1992), 269–303.
2. B. Chazelle, H. Edelsbrunner, L. Guibas, M. Sharir and J. Snoeyink, Computing a single face in an arrangement of line segments and related problems, accepted for *SIAM J. Computing.*
3. P.K. Agarwal and M. Sharir, Off-line dynamic maintenance of the width of a planar point set, *Computational Geometry, Theory and Appls* 1 (1992), 65–78.
4. D. Huttenlocher, K. Kedem and M. Sharir, The upper envelope of Voronoi surfaces and its applications, accepted for *Discrete Comput. Geom.*
5. P.K. Agarwal and M. Sharir, Applications of a new space partitioning technique, *Discrete Comput. Geom.* 9 (1992) (in press).
6. D. Halperin and M. Sharir, On disjoint concave chains in arrangements of (pseudo)lines, *Information Proc. Letters* 40 (1991), 189–192.
7. M. Katz, M. Overmars and M. Sharir, Efficient output sensitive hidden surface removal for objects with small union size, accepted for *Computational Geometry - Theory and Applications.*
8. K. Mehlhorn, M. Sharir and E. Welzl, Tail estimates for the space complexity of randomized incremental algorithms, accepted for *Computational Geometry - Theory and Applications.*
9. B. Aronov, M. Pellegrini and M. Sharir, On the zone of a surface in a hyperplane arrangement, accepted for *Discrete Comput. Geom.*
10. B. Aronov, J. Matoušek and M. Sharir, On the sum of squares of cell complexities in hyperplane arrangements, accepted for *J. Combin. Theory, Ser. A.*
11. P. Agarwal, M. Pellegrini and M. Sharir, Counting circular arc intersections, accepted for *SIAM J. Computing.*

12. S. Cappell, J.E. Goodman, J. Pach, R. Pollack, M. Sharir and R. Wenger, Common tangents and common transversals, accepted for *Adv. in Math.*

13. B. Aronov and M. Sharir, Castels in the air revisited, in preparation.

14. M. Sharir, Computational geometry, in *E.cyclopedia of Computer Science and Technology*, A. Kent and J.G. Williams (Eds.), Marcel Dekker, to appear.

15. J. Matoušek, M. Sharir and E. Welzl, A subexponential bound for linear programming and related problems, submitted to *Algorithmica.*

16. P.K. Agarwal, A. Efrat, M. Sharir and S. Toledo, Computing a segment center for a planar point set, submitted to *J. Algorithms.*

17. P.K. Agarwal, M. Sharir and S. Toledo, New applications of parametric searching in computational geometry, submitted to *J. Algorithms.*

18. Y. Karasik and M. Sharir, Optical computational geometry, submitted to *Discrete Comput. Geom.*

19. B. Chazelle, H. Edelsbrunner, L. Guibas and M. Sharir, Diameter, width, closest line pair, and parametric searching, submitted to *Discrete Comput. Geom.*

20. M. Sharir, On joints in arrangements of lines in space and related problems, submitted to *J. Comb. Theory, Ser. A.*

21. K. Kedem, M. Sharir and S. Toledo, On critical orientations in the Kedem-Sharir motion planning algorithm for a convex polygon in the plane, submitted to *Inf. Proc. Letters.*

22. N. Narendra, G. Nagy and M. Sharir, Reconstructing triangulated terrains from visibility data, in preparation.

23. B. Aronov, D.Q. Naiman, J. Pach and M. Sharir, An invariant property of balls in arrangements of hyperplanes, submitted to *Discrete Comput. Geom.*

24. A. Efrat, G. Rote and M. Sharir, On the union of fat wedges and separating a collection of segments by a line, submitted to *Computational Geometry — Theory and Applications.*

25. E. Wolfson and M. Sharir, Design and implementation of a motion planning system in 3-space, in preparation.

## Conference Proceedings

1. P.K. Agarwal, M. Sharir and S. Toledo, New applications of parametric searching in computational geometry, *Proc. 3rd ACM-SIAM Symp. on Discrete Algorithms* (1992), 72–82.

2. K. Mehlhorn, M. Sharir and E. Welzl, Tail estimates for the space complexity of randomized incremental algorithms, *Proc. 3rd ACM-SIAM Symp. on Discrete Algorithms* (1992), 89–93.

3. M. Sharir and E. Welzl, A combinatorial bound for linear programming and related problems, *Proc. Symp. on Theoretical Aspects of Computer Science* (1992), Springer Verlag Lecture Notes in Computer Science 577, pp. 569–579.

4. J. Matoušek, M. Sharir and E. Welzl, A subexponential bound for linear programming, *Proc. 8th ACM Symp. on Computational Geometry* (1992), 1–8.

5. B. Chazelle, H. Edelsbrunner, L. Guibas and M. Sharir, Diameter, width, closest line pair, and parametric searching, *Proc. 8th ACM Symp. on Computational Geometry* (1992), 120–129.

6. B. Aronov and M. Sharir, Castles in the air revisited, *Proc. 8th ACM Symp. on Computational Geometry* (1992), 146–156.

7. Y. Karasik and M. Sharir, Optical computational geometry, *Proc. 8th ACM Symp. on Computational Geometry* (1992), 232–241.

## Invited Presentations

1. M. Sharir, Complexity in arrangements: Recent developments and simple proofs, The Dagstuhl Workshop on Computational Geometry, October 1991.

Micha Sharir
Courant Institute of Mathematical Sciences, New York University
Phone: (212) 998-3375
E-mail Address: sharir@roband.nyu.edu
Contract Title: Implementation and Experimentation with Motion Planning Algorithms
Contract Number: N00014-90-J-1284
Reporting Period: 1 Oct 91 – 30 Sep 92

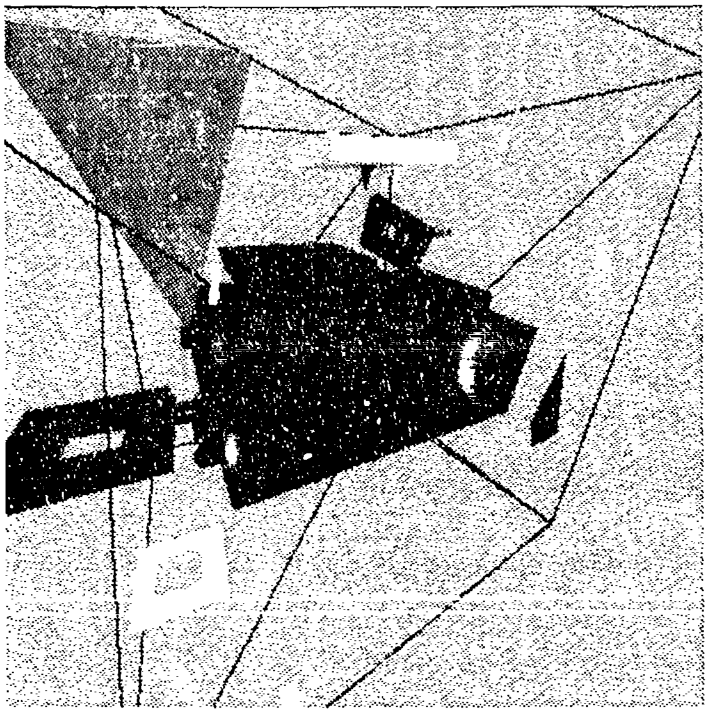
## 4 Description of Research Transitions and DoD Interactions

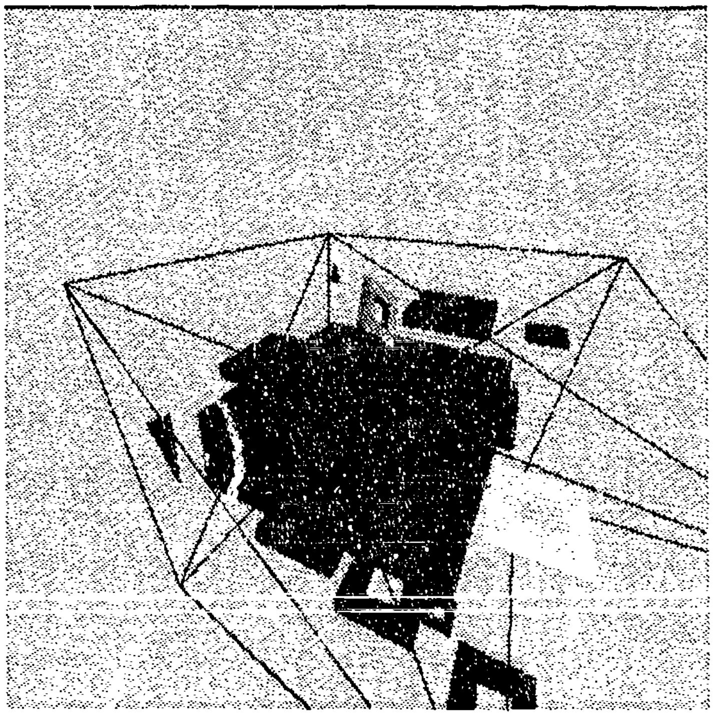
None so far.

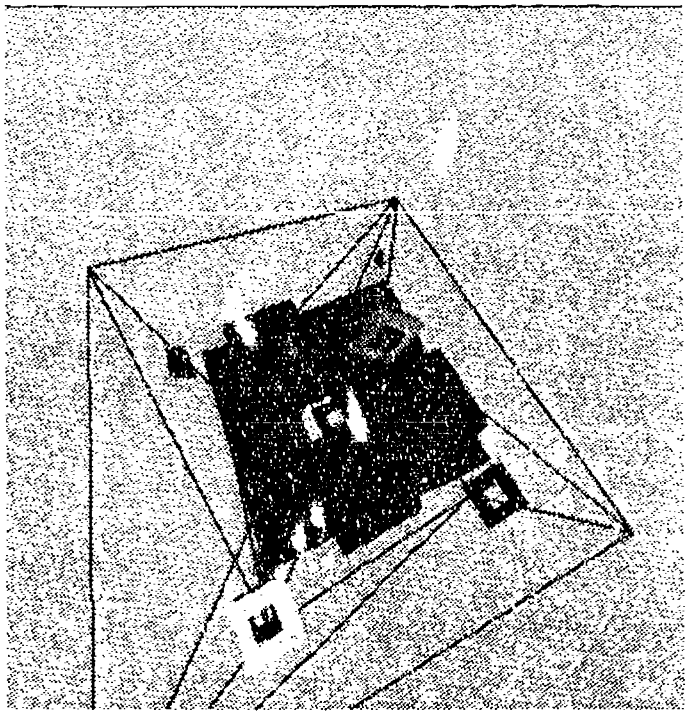
Micha Sharir
Courant Institute of Mathematical Sciences, New York University
Phone: (212) 998-3376
E-mail Address: sharir@roband.nyu.edu
Contract Title: Implementation and Experimentation with Motion Planning Algorithms
Contract Number: N00014-90-J-1284
Reporting Period: 1 Oct 91 – 30 Sep 92

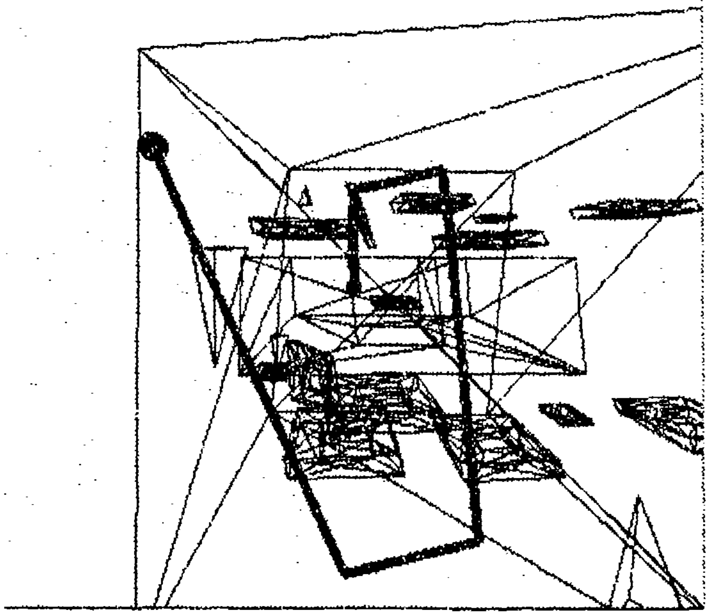
## 5 Description of Software and Hardware Prototypes

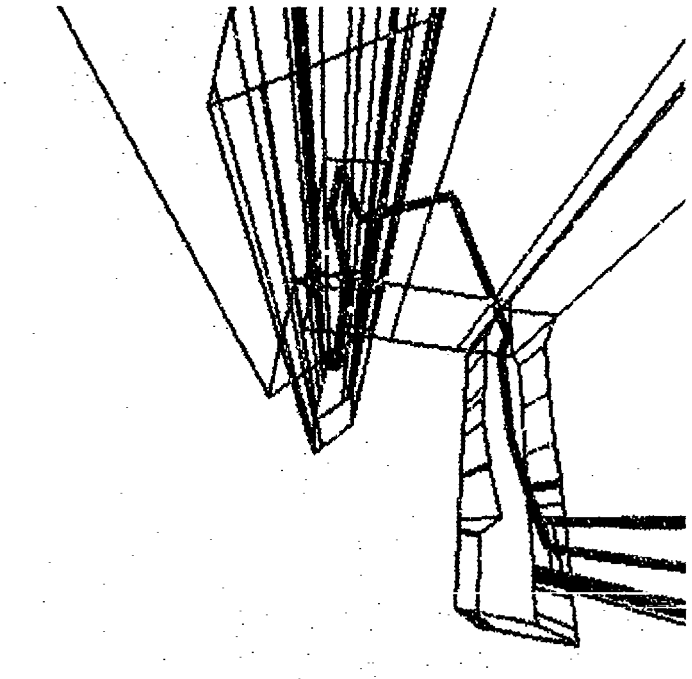
Please see Section 2 for a detailed description of the system being implemented. It is our hope that the system could be commercialized. Likely 'customers' might be the space industry (for programming flying robots), and CAD and related systems (enhancing such a system with navigation capabilities through 3-D scenes).

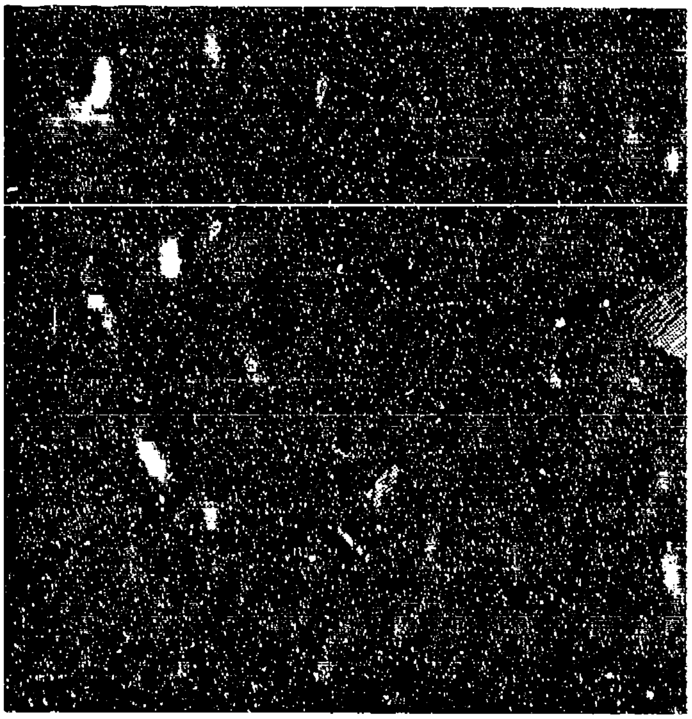












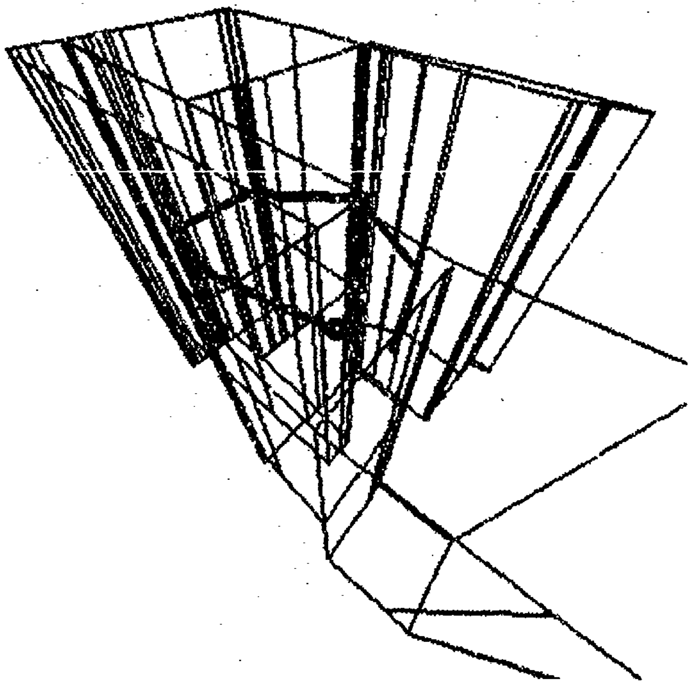


PLATE 1

Figure A:

A view of the polyhedral environment consisting of 21 polyhedral obstacles. 268 corners, 768 edges, and 512 facets.

Figure B:

Another view of the environment. The free space is decomposed into simple cells with an adjacency graph connecting them.

Figure C:

A path generated by our algorithm between two given points. After cell decomposition, the algorithm can generate a path in a fraction of a second, thus essentially in real time.

Figure D:

Another path with a skeleton view of the obstacles. A path goes to the center of each cell it traverses and out through a center point on a connecting wall to the next cell.

PLATE 2

Figure A:

A perspective view from the robot as it traverses the space along the generated path. The black and green bars represent the vertical boundary of the current cell through which the robot is passing.

Figure B:

A snapshot of a more complex path and the set of cells it still has to traverse. The path's route from the center of a common wall is quite visible here.

Figure C:

Another ride on the robot through space along a different path, showing another cell it currently traverses. In this example, there are 392 cells in the environment decomposition, which were generated in 5 seconds on a SUN Sparc II workstation.

Figure D:

A snapshot of the path and the set of cells it still has to traverse at the same moment and the same situation as in Figure C.